Digital Clock Project Report

1. Title Page

Project Title: Digital Clock

Course: Programming in C

Submitted By: Priyanshu Mehra

Roll Number: 590025830

Academic Year: 2025–26

2. Abstract

This project implements a Digital Clock using the C programming language. The primary objective is to display the current time in hours, minutes, and seconds, updating dynamically on the console.

The project demonstrates:

- Loops and modular functions
- Header files (`time.h`, `unistd.h`, `windows.h`)

- Delay functions (`Sleep()`, `usleep()`)
- Formatted output (`printf()`)

Applications include embedded systems, IoT devices, and real-time monitoring tools.

---

3. Problem Definition

Clocks are essential in both physical and digital systems. In programming, simulating a clock requires handling time values, updating them continuously, and formatting the output.

Goals:

- Display time in HH:MM:SS format
- Update every second
- Run continuously until terminated
- Demonstrate modular coding practices

Constraints:

- Accuracy of delay functions

- Portability across operating systems
- User termination handling

---

4. System Design

4.1 Algorithm

1. Start
2. Initialize time variables (hours, minutes, seconds)
3. Display time in HH:MM:SS format
4. Wait for one second
5. Increment seconds, adjust minutes/hours when needed
6. Repeat steps 3–5 continuously

4.2 Flowchart

```
┌───────────────┐
│     Start     │
└───────┬───────┘
        ↓
```

```
┌────────────────┐
│ Initialize     │
│ h, m, s        │
└─────┬──────────┘
      ↓
┌────────────────┐
│ Display Time   │
└─────┬──────────┘
      ↓
┌────────────────┐
│ Wait 1 Second  │
└─────┬──────────┘
      ↓
┌────────────────┐
│ Increment s    │
│ Adjust h, m    │
└─────┬──────────┘
      ↓
  ┌────────────┐
  │   Repeat   │
 └─────────┘
```

}

5. Implementation Details

5.1 Code Snippet

```c
#include <stdio.h>

#include <time.h>

#include <unistd.h>


void displayTime(int h, int m, int s) {

    printf("%02d:%02d:%02d\n", h, m, s);

}


int main() {

    int h = 0, m = 0, s = 0;

    while(1) {

        displayTime(h, m, s);

        sleep(1);

        s++;

        if(s == 60) { s = 0; m++; }

        if(m == 60) { m = 0; h++; }

        if(h == 24) { h = 0; }

    }

    return 0;

}
```

5.2 Concepts Used

- **Loops:** `while`, `for`
- **Functions:** Modular design (`displayTime`)
- **Header Files:** `time.h`, `unistd.h`
- **Delay Functions:** `sleep()` for Linux, `Sleep()` for Windows
- **Formatted Output:** `printf("%02d:%02d:%02d")`

---

6. Results & Sample Output

```
=== Digital Clock ===

30-11-2025

03:08:08 PM


=== Digital Clock ===

30-11-2025

03:08:09 PM


=== Digital Clock ===

30-11-2025
```

```
03:08:10 PM
```

Edge Cases:

- Midnight rollover (00:00:00)
- Noon (12:00:00 PM)
- Continuous running without lag

---

7. Conclusion & Future Work

The Digital Clock project successfully simulates a real-time clock in the console using C.

Future Enhancements:

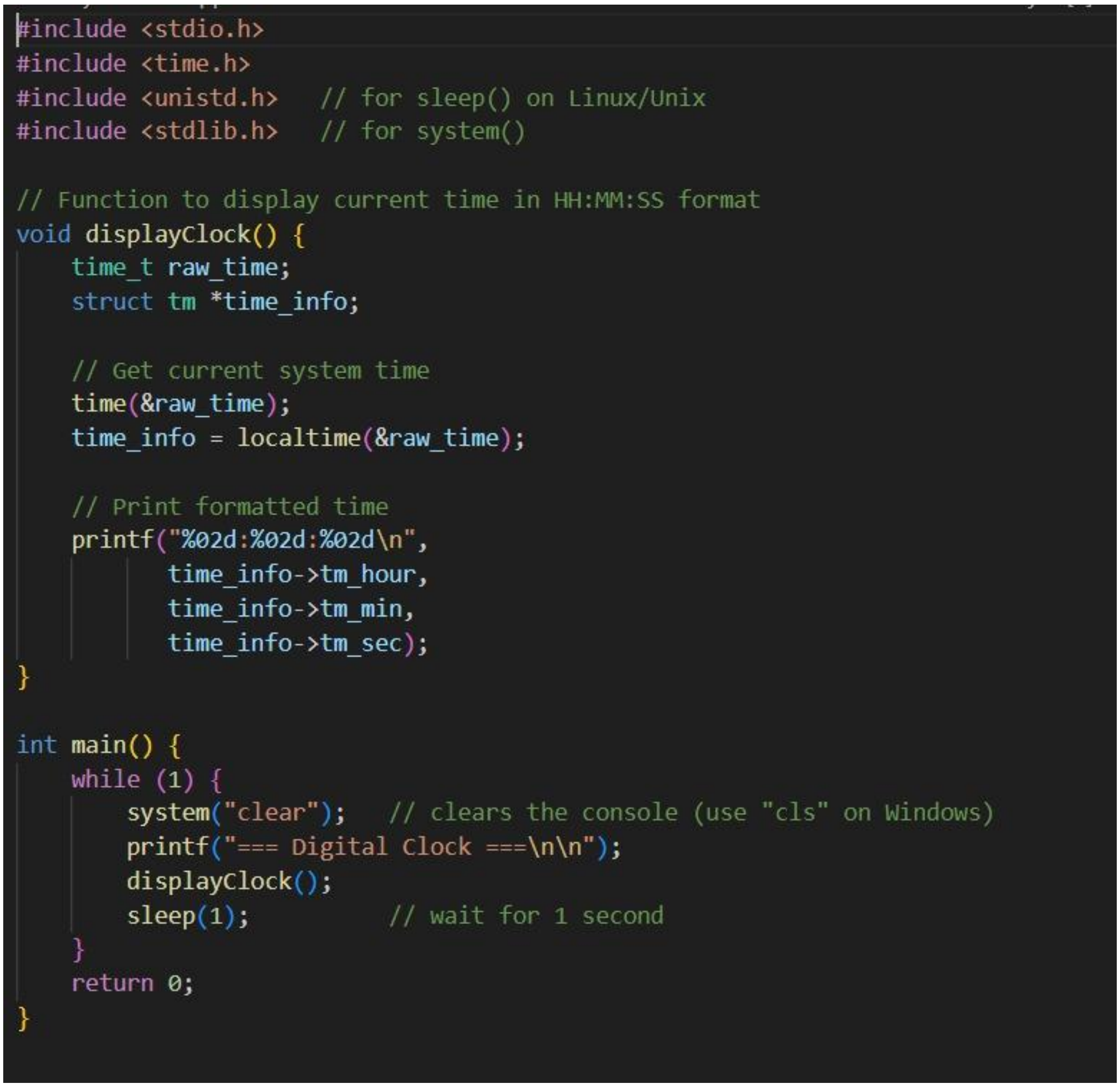
- Add alarm functionality
- Display date along with time
- Provide 12-hour/24-hour format toggle
- Build a GUI version
- Synchronize with system/NTP time
-

7. Key Concepts Used

- **Loops (Iteration)**
  - Continuous updating of time using while loops.
  - Ensures the program runs until terminated by the user.
- **Functions (Modularity)**
  - Example: displayTime() separates logic for printing formatted time.
  - Promotes reusable and clean code.
- **Header Files**
  - time.h → for handling system time.
  - unistd.h (Linux) / windows.h (Windows) → for delay functions.
- **Delay Functions**
  - sleep() (Linux) or Sleep() (Windows) → pauses execution for one second.
  - Critical for simulating real-time updates.
- **Formatted Output**
  - printf("%02d:%02d:%02d") ensures leading zeros (e.g., 03:08:09).
  - Provides user-friendly HH:MM:SS format.
- **System Time Handling**
  - Synchronization with system clock using time() and localtime().
  - Allows accurate display of current time/date.

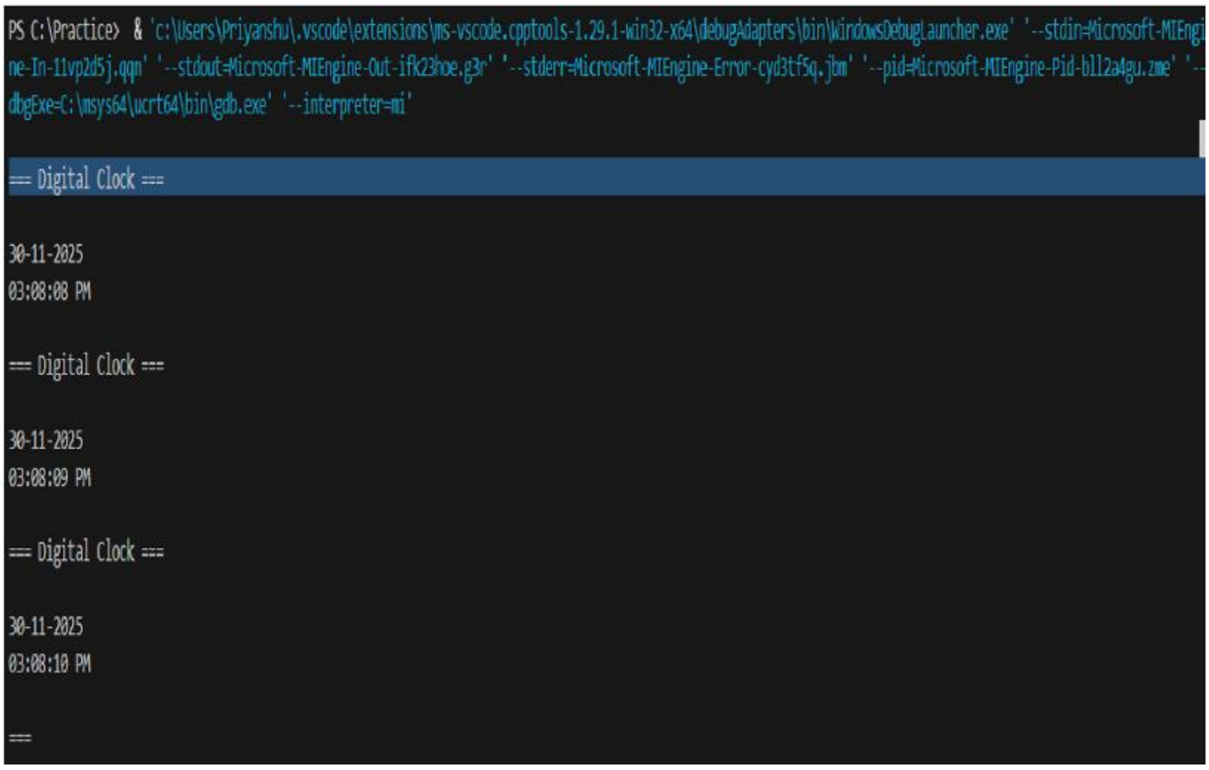
- **Real-Time Display**
  - Shows hours, minutes, and seconds updating every second.
- **Continuous Execution**
  - Runs indefinitely until user terminates (like a real clock).
- **Cross-Platform Support**
  - Works on both Windows and Linux with minor changes (Sleep() vs sleep()).
- **Modular Design**
  - Functions separate logic for display, update, and delay.
  - Easier to extend (e.g., adding alarm or date).
- **Formatted Console Output**
  - Clean, readable digital clock style.
  - Can toggle between 12-hour and 24-hour formats.
- **Future Enhancement Potential**
  - Alarm feature, GUI version, synchronization with internet time servers.
  - Date display alongside time.

CODE –



```c
#include <stdio.h>
#include <time.h>
#include <unistd.h>   // for sleep() on Linux/Unix
#include <stdlib.h>   // for system()

// Function to display current time in HH:MM:SS format
void displayClock() {
    time_t raw_time;
    struct tm *time_info;

    // Get current system time
    time(&raw_time);
    time_info = localtime(&raw_time);

    // Print formatted time
    printf("%02d:%02d:%02d\n",
           time_info->tm_hour,
           time_info->tm_min,
           time_info->tm_sec);
}

int main() {
    while (1) {
        system("clear");   // clears the console (use "cls" on Windows)
        printf("=== Digital Clock ===\n\n");
        displayClock();
        sleep(1);          // wait for 1 second
    }
    return 0;
}
```

SAMPLE OUTPUT –



```
PS C:\Practice> & 'c:\Users\Priyanshu\.vscode\extensions\ms-vscode.cpptools-1.29.1-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngi
ne-In-11vp2d5j.qqn' '--stdout=Microsoft-MIEngine-Out-ifk23hoe.g3r' '--stderr=Microsoft-MIEngine-Error-cyd3tf5q.jbm' '--pid=Microsoft-MIEngine-Pid-bll2a4gu.zme' '--
dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'

=== Digital Clock ===

30-11-2025
03:08:08 PM

=== Digital Clock ===

30-11-2025
03:08:09 PM

=== Digital Clock ===

30-11-2025
03:08:10 PM

===
```